# XML Survival Skills for DBAs

**Presented to:**


## RMOUG - Nov. 2005
## Quarterly Educational Workshop

John Jay King

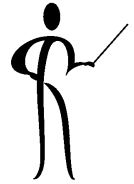
King Training Resources

john@kingtraining.com

**Download this paper and code examples from:**

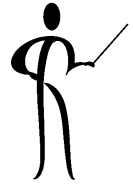
http://www.kingtraining.com

# Session Objectives



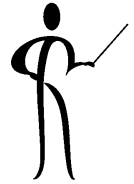
- Understand what XML extensible Markup Language (XML) and what it is not
- Use XML vocabulary effectively
- Know the rules for XML tags and "well-formed" XML documents
- Understand the relationship between XML, style sheets, DTDs, XSL, and XSLT
- Become aware of XML-supporting applications, databases, and servers

# Major Keywords



- XML
- Tag
- Element
- Document
- Well-formed XML
- Stylesheet (CSS or XSL)
- Validation (DTD and Schema)

# eXtensible Markup Language (XML)

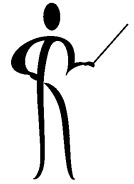


- XML is a set of rules for defining tags to describe a document's structure and parts
- XML is a "meta-markup" language, providing the syntax used to define the syntax and structure of a document, not the presentation or the format
- The XML specification is authored by the W3C (World Wide Web Consortium www.w3.org)
- "Markup" is from typesetting: publishers "markup" a document telling the typesetter how to format the page
- Computer markup languages like HTML and Troff have fixed markup features, for example, HTML provides a set of predefined "tags" used to format data (<title>,<body>,<p>,<h1>, etc…)
- XML is "extensible" because no tags are predefined
- XML is used to define your own tags or use other's tags
- All XML-based languages use the same syntax

# History of XML

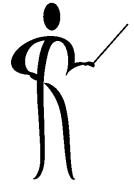
- XML is based upon markup language technology first created by IBM in the 1960's
  - Generalized Markup Language (GML) 1969 (IBM Text Description Language (TDL), renamed GML in 1971 (three developers: Charles F. Goldfarb, Ed Mosher, and Ray Lorie)
  - GML product released by IBM in 1973
  - Late 70's IBM product Document Control Facility (DCF) or "Script" introduced Document Type Descriptors (DTDs) and Document Types
  - SGML (Standard Generalized Markup Language) became a reality in the early 1980s, while powerful, SGML was deemed too complex for everyday use
- XML was created as a much simpler yet still powerful tool
- In February 1998 the Worldwide Web Consortium (W3C) proposed a recommendation for XML 1.0, XML 1.0 second edition was recommended in October 2000, XML 1.0 third edition was recommended in February 2004, XML 1.1 was also recommended in February 2004
- See  for more information about current XML (and related) standards see the website www.w3c.org or www.w3.org
- The ANSI/ISO SQLX standard provides support for XML in a database system, for more information see www.sqlx.org

# Recent Updates



- XML 1.1 has been released by W3C as a recommendation in February 2004, it provides: Unicode Control characters, line ending issue improvements, and Namespaces 1.1 incorporating corrections and a mechanism to undeclared prefixes
- XML 1.0 Release 3 was recommended in February 2004: The third edition is not a new version of XML but fixes many errors and clarifies use of keywords like must, should and may
- Extensible Stylesheet Language (XSL) Version 1.1 working draft was released in December 2003 including: Change marks, Indexes, Multiple flows, Bookmarks, and Extended support for graphics scaling, markers, and page numbers
- The XML Schema Working Group released the first public Working Draft of Requirements for XML Schema 1.1 in January 2003 adding functionality and clarifying the XML Schema Recommendations
- XPath 2.0 was recommended for release in December 2003
- Document Object Model (DOM) Level 3 is available as of December 2003; this release of DOM allows programs and scripts to dynamically update the content and style of documents, it also provides the ability to represent part of a DOM tree in memory using XPath notation

# XML and HTML



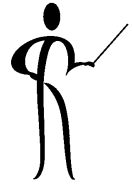
- New XML developers often confuse HTML and XML
- XML looks like HTML but is NOT HTML
- HTML specifies what each tag means and how it will appear in the browser:

```
<html>
    <body>
        <h1>Introduction to XML</h1>
    </body>
</html>
```

- XML tags describe data, interpretation varies:

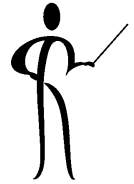
```
<myclasses>
    <class>
        <name>Introduction to XML</name>
        <author>John Jay King</author>
        <email>john@kingtraining.com</email>
    </class>
</myclasses>
```

# XML and HTML



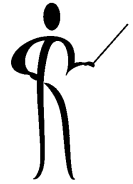
- XML rules are strict where HTML rules are often loose
- HTML tags describe the presentation of data, tags are predefined and have specific meanings to standard browsers
- XML tags describe the content of data in a document, tags are determined by the document's creator and may any valid name
- XHTML (eXtensible Hypertext Markup Language) is a hybrid of XML and HTML that provides precise HTML rules that conform to XML guidelines
  (this is the latest HTML standard too)

# XML "Family" of Software



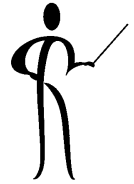
- XML 1.0 defines what tags and attributes are
- XHTML (eXtensible Hypertext Markup Language) is an XML application to replace HTML
- DTD (Document Type Definition) is a non-XML file describing the elements and syntax used by an XML document
- Schema is an XML file describing the elements and syntax used by an XML document
- XLink describes a method of using hyperlinks to reference XML
- XPointer describes hyperlinks that point to specific XML file elements
- JAXP (Java API for XML Processing) is a set of Java classes and interfaces used work with XML inside Java programs
- DOM (Document Object Model) provides access to XML document information
- SAX (Simple API for XML) provides access to XML document information
- XPath (XML Path Language) provides a mechanism for selecting XML document subsets
- XSL (Extensible Style Language) is an advanced style sheet language tailored of XML
- XSLT (XML Stylesheet Language Transformations) is used to transform XML to some other form (e.g. HTML)

# Why XML?



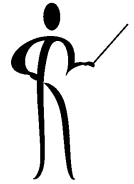
- XML is non-proprietary providing a standardized mechanism available to all vendors
- Elements and tags may be defined as needed allowing specialized languages for different disciplines
- Document templates, files, and database data can all be stored using an XML-described format
- Standardized formats make it easier to share data across various platforms, cultures, and languages
- Industry groups and companies are working to use XML to build common tag sets to allow data exchange via common data structures
- XML is frequently being used by software vendors to specify configuration information including: Databases, Web Servers, and Communications
- XML is used to describe data files used for: Word processing, Electronic Data Interchange (EDI), Sequential data storage, and many other reasons

# Some XML Languages



- XML is used for creating Markup Languages:
  - Mathematical Markup Language (MathML)
  - Synchronized Multimedia Integration Language (SMIL)
  - Voice Extensible Markup Language (VoiceXML)
  - Web Ontology Language (OWL)
  - Web Services Description Language (WSDL)
  - Speech Synthesis Markup Language (SSML)
  - A P3P Preference Exchange Language (APPEL)
  - Extensible HyperText Markup Language (XHTML)
  - XML Path Language (XPath)
  - XML Pointer Language (XPointer)
  - XML Query Language (XQuery)

# XML Tags and Elements



- XML provides the syntax necessary to define custom tags and with them custom elements
- Tags are delineated by "<" and ">" symbols:
  - Start tag <name>
  - End tag </name> (note slash)
- Empty tags incorporate the start and end together:
  - Tag <name/>
- Elements are a tag pair and their contents, as in the "name" element below
- Tags and contents together are the tag element:

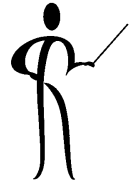
```
<name>Geoffrey Holder</name>
```

# Element Hierarchy

```
<myBooks>
    <book>          <name>Learning XML</name>
                    <author>Eric T Ray</author>
                    <publisher>O'Reilly</publisher>          </book>
    <book>          <name>XML Bible</name>
                    <author>Elliotte Rusty Harold</author>
                    <publisher>IDG Books</publisher>         </book>
    <book>          <name>XML by Example</name>
                    <author>Sean McGrath</author>
                    <publisher>Prentice-Hall</publisher>     </book>
</myBooks>
```

- The "root" element above is <myBooks> (every XML document must have a "root" element)
- Three <book> elements are part of <myBooks>
- Each <book> element includes three elements: <name>, <author>, and <publisher>
- Elements may be nested, start and end tags must entirely surround data
- Indentation is optional, but, your coworkers will thank you

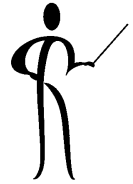
# Tag/Element Naming



- XML has specific rules for naming of Tags/Elements
- Element names must begin with a letter or an underscore
- Element names may contain letters, underscores (_), numbers, hyphens (-), and colons (:)
- Start tags must match end tags exactly
- Names in XML are case-sensitive and may not contain blanks (officially there is no limit on the length of names…)
- Names should not begin with “xml” (regardless of case)

```
<name>Jones</lastname>          incorrect
<lastname>Jones</lastname>      correct
<last name>Jones</last name>    incorrect
<lastname>Jones</lastname>      correct
<lastname>Jones</lastName>      incorrect
<lastName>Jones</lastName>      correct
```

# Attributes

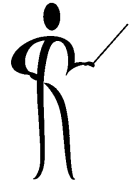


- XML elements may use descriptive attributes
- Attributes are added to an element's start tag using the name of the attribute followed by an equal sign, followed by the value of the attribute (surrounded by quotes or apostrophes)

```
<book isbn="0-13-960162-7" binding="perfect">
    <name>Learning XML</name>
    <author>Eric T Ray</author>
    <publisher>O&apos;Reilly</publisher>
</book>
```

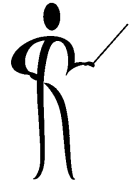
- Attribute naming rules are the same as for element naming, attribute names must be unique within an element. Usually, attributes are used to provide information about the data in an element
- Attribute values must be enclosed by quotation marks (") or apostrophes (')

# "Well-Formed" XML



- XML has a strict set of rules to determine that a document is "well-formed" or the parser will reject the file:
  - Each document must declare itself an XML document using an XML declaration (not required by all parsers today):
    ```
    <?xml version="1.0" encoding="UTF-8"?>
    ```
  - Each document must have a single "root" element that completely contains all of the other elements in the document (one set of outer tags)
  - All elements that include data must have both start <name> and end </name> tags
  - Empty tags are marked using a slash before the close of the start tag and omitting the end tag<name/> (usually include attributes)
  - Tags may not overlap, but, may be nested
  - Attribute values are surrounded by quotes (") or apostrophes (')
  - Most XML tools will refuse to process documents that are not well-formed

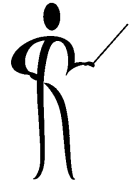
# Character Entities



- XML parsers assign specific meanings to certain characters (e.g. "<")
- XML defines five "entity references" to allow documents to include the following characters:
  - Ampersand &
  - Apostrophe '
  - Double quote "
  - Greater than >
  - Less than <
- Each entity begins with an ampersand (&) and is ended by a semicolon (;)

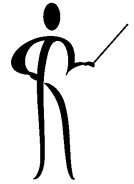
```
<company>AT&T</company>                incorrect
<company>AT&amp;T</company>            correct
```

# XML Processors



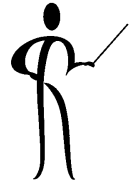
- Software that reads and does something with XML is called an "XML Processor"
- Many web browsers, XML editors, and software products are XML processors
- XML Processors typically include all or some of the following features:
  - Parser translates XML markup and data into a stream of tokens
  - Event Switcher gets tokens from parser and sorts them by function
  - Tree representation of XML document structure (may allow manipulation of nodes)
  - Tree processor that processes the XML tree for some purpose
- At the least, an XML processor reads an XML document and converts it into a form that may be used by other software; this is called "Parsing "

# Parsers



- Parsers are the fundamental part of any XML processor, they are used to:
  - Read XML data
  - Translate data into recognizable tokens (the stream of characters is separated into instructions or hierarchical information)
  - Assemble data into a hierarchy
- By design Parsers are Strict! All documents must be "well-formed" and any error aborts the parsing operation

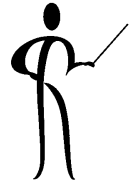
# Oracle XML Support



- Oracle's XMLtype is an Oracle-defined datatype used to store XML data within the database (CLOB underneath)
- The XML parser is part of the database
- Oracle provides several XML-oriented SQL functions to support XML use
- XML was supported weakly by Oracle8i
- Lots of XML support was added in Oracle9i and then again by Oracle 10g, check the reference manual for more information:

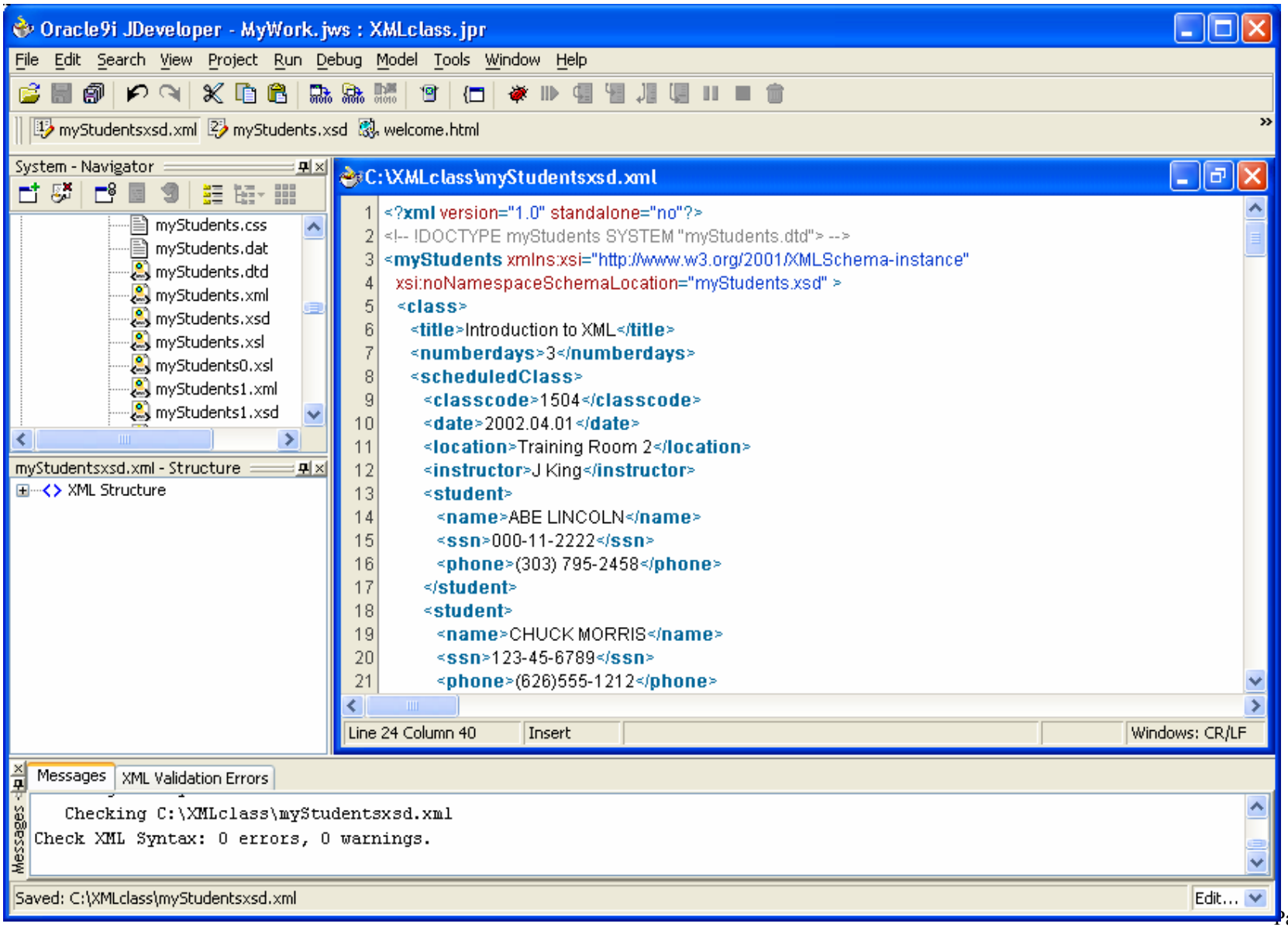
  "Oracle XML DB Developer's Guide"

# JDeveloper

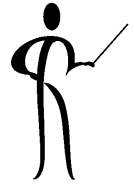


- JDeveloper10g offers support for XML file editing including:
  - File creation and manipulation
  - Verifying that a document is “well-formed”
  - Validating a document against a schema
  - XML editing supported by code-insight

# JDeveloper XML Editor

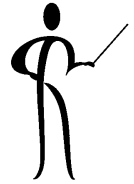


# XMLType Functions



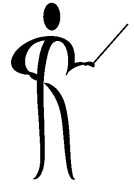
- XMLType member functions include:
  - createXML() Create XMLType instance
  - existsNode() Checks if XPath can find valid nodes
  - extract() Uses XPath to return XML fragment
  - isFragment() Checks if document is a fragment
  - getClobVal() Gets document as a CLOB
  - getStringVal() Gets value as a string
  - getNumberVal() Gets numeric value as a number

# SQL's XML Functions



- SQL provides several functions specifically for dealing with XML data including:
  - SYS_DBURIGEN(ts) - Generate DBURITYPE URL used to obtain XML data from the database
  - SYS_XMLGEN(exp) - Convert specified database row and column into an XML document
  - SYS_XMLAGG(exp) - Generate single XML document from aggregate of XML data specified by "exp"
  - XMLELEMENT(name,exp) - Generates XML element using name and exp as data
  - XMLATTRIBUTES(exp,list) - Generates XML attributes using expression
- XMLELEMENT and XMLATTRIBUTES illustrate Oracle's support of the ANSI/ISO SQLX standard

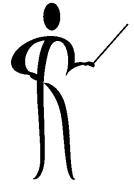
# SYS_XMLGEN



- Uses a single input expression representing a particular row/column (scalar value or user-defined type)
  - A single XML element representing scalar values is returned
  - XML elements representing each of a user-defined type's data items is returned
  - Returns an instance of SYS.XMLType data that is an XML document
- The example below uses getStringVal since SYS.XMLType data returns as CLOB and is not displayable by SQL*Plus

```
select sys_xmlgen(ename).getStringVal() Name
     from emp
     where job = 'ANALYST'
NAME
--------------------------------------------------
<?xml version="1.0"?>
 <ENAME>FORD</ENAME>
<?xml version="1.0"?>
 <ENAME>SCOTT</ENAME>
```

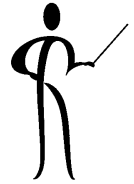
# SYS_XMLAGG



- SYS_XMLAGG aggregates all XML documents (or fragments) in an expression to produce a single document
  - ROWSET is the default tag name used
  - SYS.XMLGenFormatType may be used to change a tag name
- The example below uses the SYS_XMLGEN function to generate an XML document (example uses getClobVal since SYS.XMLType data returns as CLOB)

```
select sys_xmlagg(SYS_XMLGEN(Ename)).getClobVal() emps
            from emp
            where deptno = 10
EMPS
--------------------------------------------------
<?xml version="1.0"?>
<ROWSET>
<ENAME>KING</ENAME>
<ENAME>CLARK</ENAME>
<ENAME>MILLER</ENAME>
</ROWSET>
```

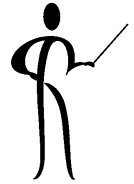
# XMLElement



- XMLELEMENT(name,exp) Generates an XML element using name and exp as data:

```
select xmlelement("employee",
       xmlelement("empid",empno),
       xmlelement("empname",ename)) myxml
    from emp

<employee> <empid>7369</empid>
    <empname>SMITH</empname> </employee>
<employee> <empid>7499</empid>
    <empname>ALLEN</empname> </employee>
<employee> <empid>7521</empid>
    <empname>WARD</empname> </employee>
<employee> <empid>7566</empid>
    <empname>JONES</empname> </employee>
<employee> <empid>7654</empid>
    <empname>MARTIN</empname> </employee>
```

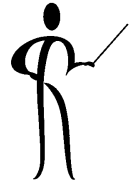
# XMLAttributes



- Generates XML attributes using an expression list:

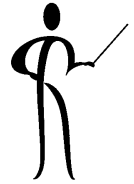
```
select xmlelement("employee",
       xmlelement("emp", xmlattributes(empno as "empno",
                         ename as "ename")),
       xmlelement("job",job),
       xmlelement("hiredate",hiredate),
       xmlelement("pay", xmlattributes(nvl(sal,0) as "sal",
                         nvl(comm,0) as "comm"))
       ) as myxml
from emp;

<employee>
  <emp empno="7782" ename="CLARK"/>
  <job>MANAGER</job>
  <hiredate>09-JUN-81</hiredate>
  <pay sal="2450" comm="0"/>
</employee>
*** More like the above ***
```

# Putting them Together



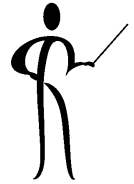
```
select xmlelement("employee",
          xmlagg(xmlelement("emp",xmlattributes(empno as "empno",
                                                ename as "ename"),
                                       xmlelement("job",job),
                                       xmlelement("hiredate",hiredate),
                                       xmlelement("pay",
                                       xmlattributes(nvl(sal,0) as "sal",
                                              nvl(comm,0) as "comm")))))
   from emp;
<employee>
   <emp empno="7839" ename="O&apos;BRIAN">
      <job>PRESIDENT</job>
      <hiredate>17-NOV-81</hiredate>
      <pay sal="5000" comm="0"/>
   </emp>
   <emp empno="7698" ename="BLAKE">
      <job>MANAGER</job>
      <hiredate>01-MAY-81</hiredate>
      <pay sal="2850" comm="0"/>
   </emp>
   *** More like above ***
</employee>
```

# Other XML Functions



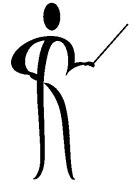
- XMLColattval
  - Creates series of XML fragments using an element name of "column" and column names and values as attributes
- XMLConcat
  - Concatenates a series of XMLType objects (opposite of XMLElement)
- XMLForest
  - Creates XML fragments from a list of arguments/parameters
- XMLSequence
  - Creates Varray of XMLType instances
- XMLTransform
  - Uses input XMLType and XSL style sheet (also XMLType) to create a new XMLType
- UpdateXML
  - Uses an XMLType and an XPATH reference and returns an updated XMLType

# PL/SQL XML INSERT



```
CREATE or REPLACE PROCEDURE insertPurchaseOrderXMLOrder IS
   PurchaseOrderXML CLOB; -- CLOB to hold XML
BEGIN
   PurchaseOrderXML :=
       <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
       <PurchaseOrder>
              <POID>1234</POID>
              <Date>2003-0401</Date>
              <CustomerID>AA1234></CustomerID>
              <Company>King Training Resources</Company>
               <!-- other elements -->
       </PurchaseOrder>';
  -- Insert the Purchase Order XML into an XMLType column
   INSERT INTO purchaseOrderTable (purchaseOrder)
       VALUES (XMLTYPE(PurchaseOrderXML));
EXCEPTION
   WHEN OTHERS THEN
       raise_application_error(-20101,
       'Error loading purchaseOrderTable, SQLCODE='||SQLERRM);
END insertPurchaseOrderXMLOrder;
```

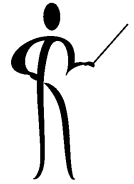
# Style Sheets



- XML describes data contents, not presentation
- Style sheets are used to provide presentation specifications for XML documents
- XML supports both CSS and XSL stylesheets
  - Cascading Style Sheets (CSS) have been supported by browsers for some time now and provides a tag-like language, but, is not XML itself
  - The eXtensible Style Sheet Language (XSL) is an XML application designed specifically for formatting XML documents! XSL uses XML syntax and more-modern browsers interpret XSL
- XML documents use a PI (Processing Instruction) to define the stylesheet type and name:

```
<?xml-stylesheet type="text/css" href="myBooks.css"?>
```

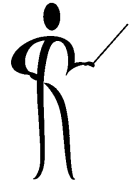
# Using XSL Stylesheets



- To use an XSL Stylesheet from an XML File the syntax is similar to when using a .css file.

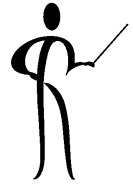
```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl"
   href="myfile.xsl" ?>
<RootyTooty>
   <!-- xml document goes here -->
</RootyTooty>
```

# XSL and CSS



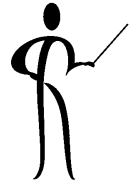
- eXtensible Stylesheet Language (XSL) is a well-formed and validated XML document providing a treasure-trove of data waiting to be used
- Unfortunately, the format of XML data is easy for computers to understand and not so easy for the average human (too many characters & tags!)
- CSS gives us some formatting capability, but, its not XML and can't take advantage of many of XML's features
- The eXtensible Style Sheet Language (XSL) was designed specifically to transform XML data from one form to another: XML document to XML document, XML document to text, Text to XML document, XML document to PDF, XML database interactions, and more!

# XSLT



- With XSL we can use eXtensible Stylesheet Language for Transformations (XSLT) to display data in different form than stored
- XSLT can also trim away unwanted portions of the XML document so that the output has only what is required
- XSLT is subset derived from XSL to use style elements for the purpose of transforming data from one format to another to describe the desired output for specific fields
- Unlike CSS which generates HTML, XSLT generates XML output
- Java programmers might use the SAX or DOM object model to read and process XML transformations, fortunately, XSL/XSLT will accomplish the same thing with the help of a browser!
- XSL and XSLT are the responsibility of the World Wide Web Consortium (W3C), please consult the W3C website for the latest information about XSL and XSLT
- XSL/XSLT provide many elements to help in the transforming or presentation of XML data
- The xsl namespace prefix is usually used when processing XSL/XSLT

# Stylesheet Elements



- Some of the Stylesheet elements include:
  - xsl:template - Creates a template "step"

```
<xsl:template match="/">
<xsl:template match="element">
```

  - xsl:apply-templates - Matches templates with input data and outputs when matches occur

```
<xsl:apply-templates
   select="element/subelement">
<xsl:apply-templates select="element">
<xsl:apply-templates />
```

# Sample Stylesheet, page 1

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
 <xsl:template match="/">
 <html> <head> <title>XML Class List</title> </head>
  <body>
    <h1 align="center">XML Class List</h1>
    <table>
    <xsl:apply-templates select="myStudents/class"/>
    </table>
  </body>
 </html>
 </xsl:template>
 <xsl:template match="class">
    <tr> <td> <h2><xsl:value-of select="title"/></h2> </td>
  </tr>
    <tr> <td>
      <h2>Number of days = <xsl:value-of
  select="numberdays"/></h2>
      </td> </tr>
   <xsl:apply-templates select="scheduledClass"/>
 </xsl:template>
```

# Sample Stylesheet, page 2

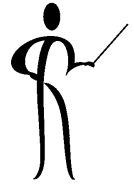
```
 <xsl:template match="scheduledClass">
  <tr>  <td>  Class code = <xsl:value-of select="classcode"/>
            Date = <xsl:value-of select="date"/>  </td>  </tr>
  <tr>  <td> Location = <xsl:value-of select="location"/>
            Instructor = <xsl:value-of select="instructor"/>
             </td> </tr>
  <xsl:apply-templates select="student"/>
 </xsl:template>
 <xsl:template match="student">
  <tr> <td> <xsl:value-of select="name"/> </td> </tr>
  <xsl:apply-templates />
 </xsl:template>
</xsl:stylesheet>
```

- The statement select="student" locates a specific node and apply-templates select="myStudents/class" matches up with the node xsl:template match="class"

# DTDs & Schemas: "Valid" Documents

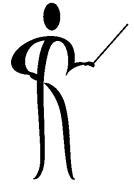
- To be well-formed:
  - Documents must include an XML declaration
  - Root element appears only once
  - Each start tag has a matching end tag
  - Elements may not overlap
- To be well-formed may not be enough, what if you want to share your XML with others?
- What guarantees that your elements will match those used by others?
- XML provides two mechanisms for validating XML files, Document Type Definitions (DTDs) and Schemas

# Document Tag Definition (DTD)



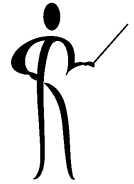
- Document Type Definitions (DTDs) specify the elements a document must contain, the element sequence, and the contents of each element
- Schemas (discussed later) are also used for validation purposes
- The DTD was the first mechanism used in XML to ensure that document definitions matched
- Using a common definition means that team members may collaborate by merging documents
- Creating DTDs can be complex, DTD syntax is different from standard XML
- Using DTDs ensures that XML documents follow rules
- DTDs may be specified internally inside the XML file or externally in a separate file
- XML tags are allowed to be just about anything the document creator feels is reasonable, when passing XML documents it seems like a good idea for the recipient of a document to have a mechanism for validating the XML

# DTD Rules



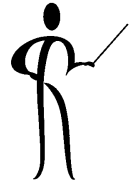
- A DTD is responsible for:
  - Naming document type
  - Defining each element a document might use
  - Defining the data type of each element
  - Defining the number of occurrences allowed for each element (zero, one, many)
  - Defining attributes used for each element and the allowed attribute values
- XML Schema definitions (recommended by W3C in May 2001) are intended to replace DTD use eventually (XML Schemas are written using XML).

# DTD Syntax



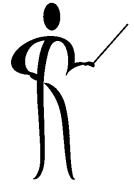
- DTDs include many features including:
  - Document type: name of the document
  - Elements: fields in the document
  - Data type: PCDATA (parsed character) most common (tags will be interpreted)
  - Field occurrences:
    - Plus-sign (+) - one or more
    - Asterisk (*) - zero or more
    - Question mark (?) - zero or one
    - Default - exactly one
  - Attributes for a field and permissible values may be specified
  - DTDs may use standard XML comments <!-- comment -->
- DOCTYPE is the high-level (root) tag in a DTD, it includes the DTD text or reference a URI that points to a file containing the DTD
- If a DTD is included in the same file as the XML document, the DOCTYPE and associated specifications are included at the top of the file (might use standalone="yes" if no external files are used)

# Sample DTD



```
<?xml version="1.0"?>
 <!ELEMENT myStudents (class+)>
 <!ELEMENT class (title, numberdays, scheduledClass+)>
 <!ELEMENT title (#PCDATA)> <!ELEMENT numberdays (#PCDATA)>
 <!ELEMENT scheduledClass
   (classcode,date,location,instructor,student*)>
 <!ELEMENT classcode (#PCDATA)> <!ELEMENT date (#PCDATA)>
 <!ELEMENT location (#PCDATA)> <!ELEMENT instructor (#PCDATA)>
 <!ELEMENT student (name)> <!ELEMENT name (#PCDATA)>
```

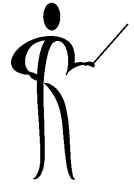
# Using XML DTDs



- To reference an XML DTD from an XML document, use a PI as follows:

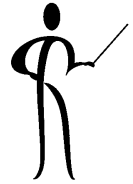
```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE myStudents SYSTEM "myStudents.dtd">
<myStudents>
    <class>
        <title>Introduction to XML</title>
    ...
```

# Problems with DTDs



- DTDs have their own syntax, different from standard XML
- All DTD elements are global in nature
- DTD's have no mechanism for specifying the type of data that belongs in a field

# Schemas



- W3C has created a new, improved method for validating XML documents called a Schema
- Schemas are well-formed XML documents themselves that describe the XML document's format
- With Schemas, XML documents and their format descriptions use the same basic formatting rules (XML) perhaps making it easier to work with both
- Schemas are used for XML document validation
- Schemas are also useful as documentation tools, since they follow the rigid XML standard they are machine-readable!
- As members of the XML family are updated (e.g. XPath, XSLT, XQuery), Schemas are incorporated into their design

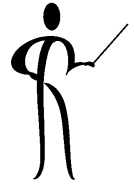
# XML Schema Syntax, page 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">
   <xs:element name="myStudents">
       <xs:complexType>
             <xs:sequence>
                <xs:element ref="class"
                          maxOccurs="unbounded"/>
             </xs:sequence>
       </xs:complexType>
   </xs:element>
   <xs:element name="class">
       <xs:complexType>
             <xs:sequence>
                   <xs:element ref="title"/>
                   <xs:element ref="numberdays"/>
                   <xs:element ref="scheduledClass"
                          maxOccurs="unbounded"/>
             </xs:sequence>
       </xs:complexType>
   </xs:element>
```

# XML Schema Syntax, page 2

```
<xs:element name="title" type="xs:string"/>
  <xs:element name="numberdays" type="xs:byte"/>
                    ...
  <xs:element name="classcode">
      <xs:simpleType>
            <xs:restriction base="xs:short">
                  <xs:enumeration value="1504"/>
                  <xs:enumeration value="1508"/>
                  <xs:enumeration value="1511"/>
            </xs:restriction>
      </xs:simpleType>
  ...
</xs:schema>
```

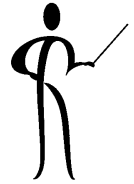
# Using XML Schemas



- To reference an XML Schema from an XML document, modify the root element to include the schema:

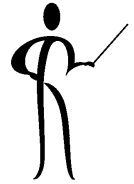
```
<?xml version="1.0" standalone="no" ?>
<?xml-stylesheet href="myStudents.css" type="text/css" ?>
<myStudents
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance“
    xsi:noNamespaceSchemaLocation="myStudents.xsd">
    <class>
          <title>Introduction to XML</title>
    ...
```

# XPath and Xlink



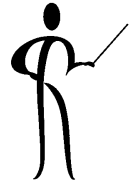
- XML Path Language (XPath) is designed to provide quick and easy access to any node in our document's hierarchy
- Having lots of information in XML is of little use if we cannot get to data when it is needed, XPath to the rescue!
- XPath provides a mechanism to address any element or attribute
- XLink (XML Linking Language) provides a hyperlink-type capabilty to XML
- The XLink specification is in a state of flux and is not supported by many browsers and tools, it is most useful from within programs
- XLink allows links to be constructed using any element (XML does not have any pre-defined element tags)
- XLinks build on the capability of HTML linking and avoid some of the problems

# Oracle9i R2 – XML DB



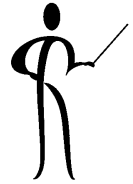
- XML-related features have been greatly enhanced in Oracle9iR2
- XML DB allows native XML database activity in addition to the standard ORDBMS functionality
- XML DB allows XML documents to be stored in the database in two ways:
  - Unstructured XMLType (CLOB); XML stored as string of bytes in the database
  - Structured XMLType with document “shredded” into objects in the ORDBMS (SQL ’99 standard objects)
- Oracle supports XPath notation to access data within a document (though not fully, coming in future releases)

# Oracle 10g R1 – XML DB



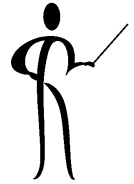
- Once again, XML-related features have been greatly enhanced in Oracle 10g
- IMPORT/EXPORT utility may load XML data into XMLType tables
- Registered Schemas may be modified (evolved) using a new PL/SQL packaged procedure DBMS_XMLSCHEMA.CopyEvolve()
- DBMS_XMLGEN now supports Hierarchical Queries (CONNECT BY) and allows "pretty printing" to be turned off
- Globalized character encoding and multibyte characters may be used, also client character set may be different from the database character set
- C and C++ DOM API for XML (both XML DB and XDK)
- SQL*Loader loads "structured" or "unstructured" XMLType Tables and Columns
- Oracle Text new XML features:
  - CTXXPATH now supports postional predicates and attribute existence expressions
  - Oracle Text supports highlighting for INPATH and HASPATH operators of ConText
  - New syntax for XPath ora:contains function
- XMLType may be an Advanced Queuing (AQ) payload type
- See the Oracle documention:
  Oracle XML DB Developer's Guide
  Oracle XML Developer's Kit Programmer's Guide
  Oracle XML API Reference

# Oracle 10g R2 – XML DB



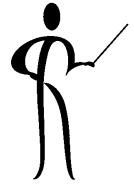
- Full support for XMLtype in Java, C, and C++
- XSLT 2.0 with XPath functions and operators
- JAXB Compiler
- XQuery: XMLQUERY and XMLTABLE functions
  - XMLQuery Build XML data, query both XML and relational data using XQuery
  - XMLTable Create relational tables and columns from XQuery results
- XPath Rewrite speeds up XML queries
- InsertXML(), AppendChildXML(), InsertXMLBefore(), and DeleteXML() functions added to UpdateXML()
- SOAP Services in C and C++
- SQL/XML (SQLX) 2003 standard: XMLPI, XMLComment, XMLRoot, XMLSerialize, XMLCDATA, and XMLParse
- XML Object support in Enterprise Manager Web Console
- HTTPS support for XML DB
- Oracle XDK PL/SQL packages deprecated: XMLDOM, XMLPARSER, XSL_PROCESSOR replaced by DBMS_XMLDOM, DBMS_XMLPARSER, DBMS_XSLPROCESSOR

# Structured Storage



- XML document is “shredded” into database objects
- Documents must conform to a registered XMLSchema; XML DB will use the XML Schema to generate SQL
- Structured Storage has several advantages:
  - Memory management is better than with CLOB
  - Storage requirements are reduced
  - Indexing is possible
  - Partial or in-place updates are possible
- Adding and retrieving XML documents to the database is slower when using Structured Storage

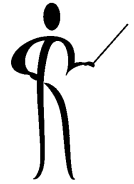
# Schema Validation



- XML Schemas are powerful, but, do not provide some features we take for granted in the database such as UNIQUE key and FOREIGN key constraints
- By default, Oracle does not completely validate documents as they are inserted into the database; restriction facets such as minLength, maxLength, and patterns are ignored

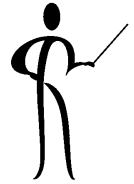
  Schema validation may be enabled for individual schemas  via Check Constraints or Triggers

# XML DB Components



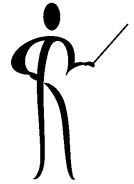
- XMLType — Data type defining the column or table as XML data and including methods to allow operations on the XML such as XSL transformations and validation via XML Schema
- XMLSchema — Complete XML Schemas may be registered with XML DB to validate documents and to define how documents will be stored by the database
- XML DB Repository — Provides mechanism for associated URIs with XPath notation to access XML data; supports interaction with HTTP, FTP, WebDAV clients
- SQL/XML (SQLX) — XML DB includes many operators that are part of the new ANSI/ISO SQL/XML (SQLX) standard to:
  - Query and access XML documents as part of SQL
  - Generate XML using the SQL SELECT statement

# What is WebDAV?



- WebDAV is an IETF (www.ietf.org) standard set of HTTP extensions allowing an HTTP Server to serve files to a WebDAV-enabled client
- Any WebDAV-enabled product can read and update XML content stored in the XML DB Repository
- Since both Microsoft Office (Office XP and beyond) and Oracle support WebDAV, they work together automatically
- Some other WebDAV-enabled products: Microsoft Internet Explorer, Altova XMLSpy, Macromedia MX and others
- XML's promise of portable data is greatly facilitated by WebDAV

# Running Oracle's XDB Demo

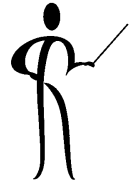


- Go to http://technet.oracle.com

  If you don't already belong:
  - No cost (other than an email address and occasional spam)
  - Most trial software available for download
  - Many white papers and demos

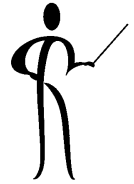
- Look under "XML" for the "XDBBasicDemo"
  - Download it
  - Download other software XML editor, FTP package (not required, but makes installation of demo easier)
  - Try WebDAV by following the wll-laid-out instructions

# XML and JSP (User Tag Libraries)



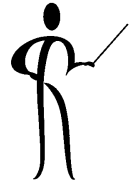
- JSP developers write HTML and enclose Java code in JSP tags
- JSP tags are coded into HTML and start with "<%" and end with " %>“
- Separating Java and HTML makes code more useful to web designers
- JSP Custom Tags allow elimination of most Java code from JSPs
- Custom Tags are probably the most powerful feature of JSPs
  - Web Page designers may concentrate on using tags to create functional web sites
  - Developers may concentrate on the nitty-gritty details necessary to make the tags work
- A few simple steps are required to create and use Custom Tags:
  - Create Java class to provide the low-end code
  - Create a Tag Library Descriptor (TLD) connecting the .class file (or .jar file) to a tag name
  - Reference and use the TLD
  - Use Taglib to connect TLD to a prefix used in the JSP
  - Use prefix and tag name in JSP source

# XML and Java (DOM, SAX, SOAP)



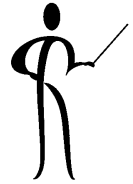
- Java is relatively new (1995) promising portable code; write-once, run-everwhere
- XML's text base and standardization provide portable data; store-once, use-everywhere
- XML and Java seem made for each other!
- Programs use XML via Application Programming Interfaces (APIs)
- Low-level APIs allow programmers to deal with the XML document and its data
  - DOM, SAX, and JDOM are the most commonly-used low-level APIs
  - JAXP (Java API for XML Programming) is relatively new and is becoming popular
- High-level APIs provide a simpler interface that calls one of the lower-level APIs "under-the-covers"
  - High-level APIs tend to be easier to develop with but usually add processing costs (no free-lunch!)
  - XML data binding is an example of a high-level interface

# Programmer APIs



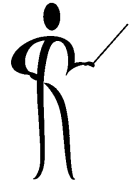
- DOM (Document Object Model) has been around for many years and is frequently used
- SAX (Simple API for XML) is newer than DOM and offers many Java-specific features
- JDOM (Java Document Object Model) is a Java-specific API tailored specifically to the needs of Java programmers
- JAXP (Java API for XML Programming) is really a higher-level API designed to take some of the complexity out of using DOM, SAX, or JDOM

# DBA's XML role



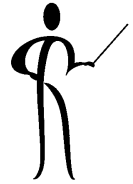
- The DBA's role in XML development, deployment, and execution is crucial
- Database access must be granted to applications and database object using XML must be created
- DBAs are the "answer-people" to whom programmers and users bring any manner of programming or SQL problem remotely connected to the database, so, XML is one more tool to be aware of
- DBAs must make sure that XML developers access the database as efficiently as possible and help create designs that provide reasonable performance
- Finally, most Oracle and other databases are shipping configuration files in XML format and DBAs must be able to understand the syntax in order to make the appropriate modifications

# Wrapping it all Up



- XML provides portable data to compliment the portable programming that Java provides
- DBAs must become aware of the basics of XML and its use
- XML is increasingly being used for data interchange and configuration
- As developers begin to use XML DBAs must be ready to support and improve their use of XML

# XML in "The Real World"



- Okay, what will we use XML for?
  - Passing data between computer systems
    - Transaction data (new EDI standards)
    - SOAP, WSDL, UDDI, and WSIL (web services)
    - Transmission/exchange of data in universal format
  - Control files (database/server/software configuration)
- XML is not intended to be a replacement for database management systems
  - Silly things I've heard smart people say…
  - Database management systems like Oracle increasingly hold non-standard data (photos, sound, etc…)
  - Oracle and other databases now provide the ability to:
    - Produce query output in XML form
    - Store XML data in its native form (unstructured data)
    - Parse XML data and store it in relational form (structured data)


# Training Days 2006

**Mark your calendar for:**

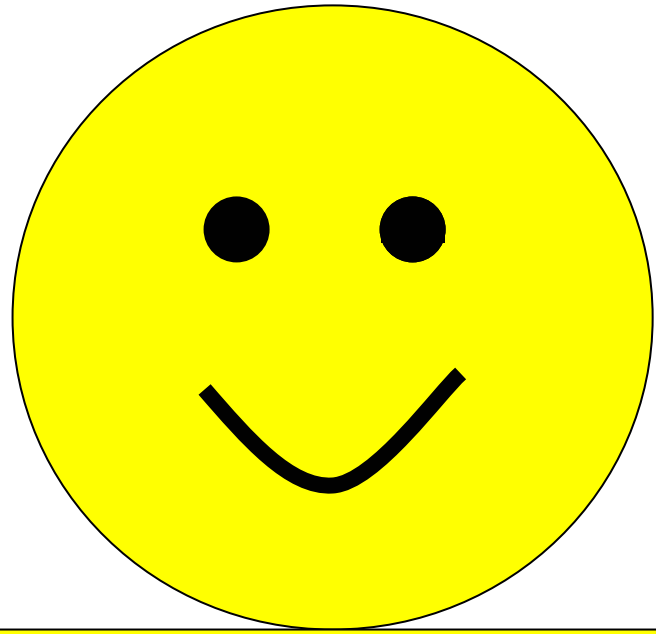
# February 15-16, 2006!

# To contact the author:

John King

King Training Resources
6341 South Williams Street
Littleton, CO 80121-2627 USA
1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com



Thanks for your attention!

## Today's slides and examples are on the web:

http://www.kingtraining.com